Colour DOS

A guidline for the Disk Operating System
of the EACA EG2000 Colour Genie Computer

Table of Contents.

1. Introduction

This document contains helpful information about the functioning and a more extensive use of DOS on the EG2000 Colour Genie computer. It gives an overview of the details of DOS, and how to access DOS from assembly language programs. Some program examples are included.

2. Floppy disk composition

This chapter gives an overview of the way that floppy disks are formatted and how the data on the floppy disk is distributed and managened.

The smallest data unit on a floppy disk is one sector. One sector always contains 256 bytes. Sectors are numbered. Machine programs can access sectors directly for either read or write operations.

Every 5 consecutive sectors form a granule (1280 bytes). In a normal floppy disk composition in files, that are addressed with their filename, the granule is the smallest addressable unit. This means that every file occupies a certain number of granules. This can lead to the situation that DOS reports a floppy disk that contains a lot of small files as full, although the majority of sectors is unused.

Another unit that is important in the use of DOS, is the lump. The size of a lump depends on the floppy disk format:

Floppy disk format	Granules per lump
A,E,I (SS,SD)	2
B,F,J (DS,SD)	4
C,G,K (SS,DD)	3
D,H,L (DS,DD)	6

During formatting, 2 system files a written onto the floppy disk. These system files are hidden, and are not listed using the CMD"l" command:

DIR/SYS
This file contains the directory of the floppy disk. This is the only file that contains protected sectors. This means, that when reading this file, an error 6 is reported. This serves as an extra protection and recognition of this file. DIR/SYS differs in length depending on the floppy disk format.

NCW1983/JHL
This file always occupies the first granule (sector 0 to 4) on the floppy disk. Its main task is to make the floppy disks compatible with the operating systems for the TRS 80 and Video Genie computers (there it is called BOOT/SYS and it contains the load routine for DOS). On the Colour Genie, only the third byte of the first sector is important; it indicates, in which lump on the floppy disk, the directory starts.

For studying the floppy disk composition, a program like Colour-Zap is strongly recommended. Such a program allows you to look at the sectors on the floppy disk.

The first sector of DIR/SYS contains information about the floppy disk. The bytes 00H to 05H indicate the free granules on the floppy disk. Every byte corresponds with one lump. Starting with bit 0, only that number of bits is used, that corresponds with the number of granules per lump. So, bit 0 of the first byte indicates whether the first granule on the floppy disk is occupied (this granule is always occupied by NCW1983/SYS). If a granule is occupied, the corresponding bit is set to 1. Unused bits are always set to 1. A completely used lump has its corresponding byte set to FFH.
The bytes 60H to BFH are not used by the Colour Genie. They are used in some TRS80/Video Genie operating systems to indicate defective granules.
The bytes D0H to D7H contain name, the bytes D8H to DFH contain the date of the floppy disk.
The remaining bytes of the first sector have no meaning on the Colour Genie.

The second sector of DIR/SYS contains the hash code table for several files. Here, for every file, a hash code from the name is stored to speed up the search of a file. The position of a hash code in the second sector indicates, where the real entry in DIR/SYS is located. When one would imagine this sector as a table with 8 rows and 32 columns, then every column contains the hash codes for one of the next sectors. Column 0 for the

third sector, column 1 for the fourth sector etc.. When a byte is 0, it means that there is no file entry at the corresponding location.
Byte 1FH of the second sector has a special meaning. It indicated the length of DIR/SYS-10. So, when is contains 5 (as with SS/DD floppy disks), DIR/SYS is occupies 15 sectors.

The next sectors contain the file entries for the files present. Every file uses a block of 32 bytes, one sector can hold up to 8 files. There are 2 types of entries:

The first (and in most cases only) entry for a file and its continuation (necessary for long files, when the file is stored on different locations on the floppy disk). The bytes of the first entry contain following information:

Byte 0:

Bit	Meaning when set
7	Always 0
6	System file
5	No meaning
4	Entry in use, contains a file
3	Hidden file, not listed using CMD"l"
2-0	Always 0 on the Colour Genie

Byte 1:

Bit	Meaning when set
7	Always 0
6	System file
5-0	No meaning

Byte 2: No meaning
Byte 3: Indicates which byte in the last sector of the file does not belong to the file anymore. Here the rule is 0=256.
Byte 4: Not used by DOS
Bytes 5-12: Contain the file name, padded with spaces.
Bytes 13-15 Contain the file extension.
Bytes 16-19: Always 0 on the Colour Genie
Bytes 13-15 Indicate how many sectors are used by this file.

The remaining bytes indicate the location of the data. It is possible that the file is divided into several blocks. For this reason, 4 pairs of bytes are reserved. If they are not sufficient, the file will obtain a second entry.
When the first byte of a pair equals FFH, the file has no more following blocks. The remaining byte pairs are meaningless.
When the first byte of a pair equals FEH, the file has an additional entry in the directory. The second byte indicates the location of the entry; the bits 0-4 show in which sector, and the bits 5-7 indicate which entry within the sector.
Otherwise, the first byte shows in which lump the data block belonging to the file starts. Bits 5-7 in the second byte then show, in which granule of this lump the data block starts, and the bits 0-4 indicate the length of the data block (in granules).

An additional entry of a file can be recognised by bit 7 of byte 0. Whenever this bit is set to 1, then byte 1 indicates the location of the previous entry (see above). Bytes 2-12 stay unused, and the bytes 23-31 have the same meaning as described above.

The separation of files into several data blocks may look somewhat complicated, but it allows to use of every free granule on the floppy disk.

Deleting a file is done by setting the first byte of the file entry and the hash code to zero. The disk space occupied by this file is then released. The data and all file entries remain unchanged.

3. File formats

This chapter gives an overview of the way that programs are stored on the floppy disk.

BASIC programs starts with a byte set to FFH. After that follows the program as it is stored in memory.

BASIC programs saved with the SAVE" ",A option, contain the ASCII characters, that are also shown by the LIST command.

Pascal source files created with the Colour Pascal 2.0 compiler are also stored as a text file containing the ASCII characters of the source file.

Machine language programs have a more complex format:
They are divided into blocks of 256 bytes maximum. Every block contains 4 extra bytes of load information:

Byte	Meaning
1	1 = Load data block. 2 = Last data block. 3 = Ignore data block.
2	Length+2: 2 means 256 bytes, 1 means 255 bytes, etc..
3-4	Load address of the data block

The last block of a file has only 4 bytes, 2 times 2 followed by the entry address of the program.

4. The File Control Block (FCB)

This chapter gives information about the FCB structure and the usage of an FCB.

To every opened file, a File Control Block is assigned. This FCB is the only connection to the file on the floppy disk. The computer does not store if and how many files are opened, but installs a FCB. All operations use and go through the FCB.

When using DOS routines, the address of the FCB used must be stored in register DE. Also, a buffer of 256 bytes is needed for every file, to read data into. The address of this buffer must be indicated when the file is opened.

The programmer indicates, when opening a file, where the FCB is located in memory. This requires a 32 byte memory area, that must be reserved for this purpose only.

Normally, DOS routines manage the FCB themselves; the programmer has just to call the right routines. It is however possible, to change the FCB for own purposes. The meaning of the bytes inside the FCB are as follows:

Byte 0:

Bit	Meaning when set
7	File opened
6-2	No meaning
1	File uses the complete floppy disk
0	Sectors are written protected (only DIR/SYS)

Byte 1:

Bit	Meaning when set
7	Always 0
6	System file
5	Buffer contains data of the next sector
4	Buffer contains data that still has to be written
3	Always set on the Colour Genie
2-0	Always 0 on the Colour Genie

Byte 2:	Identical to byte 1 of the directory entry
Byte 3-4:	Contain the address of the buffer
Byte 5:	Indicates which byte of the current sector will be read/written next
Byte 6:	Contains the used drive number
Byte 7:	Contains the corresponding directory entry position
Byte 8:	Identical to byte 3 of the directory entry
Byte 9:	Contains the record length (0=256 bytes)
Bytes 10-11:	Indicate the current sector that is processed
Bytes 12-21:	Correspond with the bytes 20-29 of the directory entry
Bytes 22-23	FFH, if the file has only one directory entry. Else, they indicate the location of the second directory entry.
Bytes 24-31:	Correspond with the bytes 22-29 of the second directory entry (if present)

For the use of most DOS routines, the address of the used FCB must be loaded into register pair DE. After returning, the zero flag is cleared in case no error occurred. If an error occurred, register A contains the error number. The error code is identical to the DISK-Error messages.

5. DOS routines

A list of the DOS routines, with a short description will follow. First, the register values that are expected are indicated. After that, the return values are given. If nothing else is mentioned, the routine will not change any registers except AF.

File Open : CE24H

Entry:

DE : FCB address
HL : Buffer address
B : Record length
in FCB : Filename

Return:

AF : Error code
in FCB : see FCB

This routine opens an existing file on the floppy disk. The filename in the FCB must follow the usual conventions and must end with a 03 or 0DH. HL points to a buffer with a minimum size of 256 bytes.
Register B indicates the record length. This means how many bytes are processed at a read or write operation. With B = 0, data are processed sector wise, so with 256 bytes at a time. All values from 0 to 255 are valid.

File Initialise : CE20H

Entry:

DE : FCB address
HL : Buffer address
B : Record length
in FCB : Filename

Return:

AF : Error code
in FCB : see FCB

This routine is similar to the file open, which is called first. When the file however does not exist, the file is created on the floppy disk. Therefor, this routine should only be used for write operations.

Read and write of single bytes.

For reading / writing single bytes, special easy to use routines are available. When opening the file, the record length must be put into register B.

Read one byte : 0013H

Entry:

DE : FCB address

Return:

AF : Byte read, or error code when zero flag is clear.

Write one byte : 001BH

Entry:

DE : FCB address
A : Byte to be written

Return:

AF : Error code

Read one record : CE36H

Entry:

DE : FCB address
HL : When record length = 0 : buffer address

Return:

AF : Error code
in buffer: Data read.

This routine reads one record. The record length is determined when the file is opened. If one works sector wise, the sector that is read is stored in the buffer that is indicated when the file is opened. If the record length = 0, the record is stored into a second buffer; register pair HL contains the address of this buffer.

Write on record : CE39H or CE3CH

Entry:

DE : FCB address
HL : When record length = 0 : buffer address

Return:

AF : Error code

This routine writes one record. The record length is determined when the file is opened. If one works sector wise, the data must be present in the buffer that is indicated when the file is opened. Otherwise, the data must be present in a second buffer; register pair HL contains the address of this buffer. Routine CE3CH also verifies the data written.

File Close : CE28H

Entry:

DE : FCB address

Return:

AF : Error code

This routine closes a file a writes all necessary information onto the floppy disk. This routine must only be called, when the file has changed due to write operations.

Random Access Routines.

The following routines enable you to change the NEXT pointer that determines which record has to be read or written next.

Set NEXT to BC : CE42H

Entry:

DE : FCB address
BC : Record number

Return:

AF : Error code

This routine puts the NEXT pointer on the record indicated by the record number stored in register pair BC.

Set NEXT to 0 : CE3FH

Entry:

DE : FCB address

Return:

AF : Error code

This routine puts the NEXT pointer to the start of the file.

Set NEXT to EOF : CE48H

Entry:

DE : FCB address

Return:

AF : Error code

This routine puts the NEXT pointer to the end of file (only useful for writing).

Set NEXT to byte address : CE4EH

Entry:

DE : FCB address
HLC : Byte address

Return:

AF : Error code

This routine puts the NEXT pointer onto a byte, addressed by the 3 byte address stored into the registers HLC. This routine is useful when working with the byte routines 0013H and 001BH.

Decrement NEXT : CE45H

Entry:

DE : FCB address

Return:

AF : Error code

This routine decrement the NEXT pointer; the record last processed is used again.

Check Filename and copy into FCB : CE1CH

Entry:

DE : FCB address
HL : Address of filename

Return:

AF : Error code

This routine checks the indicated filename. If it complies to the rules, it will copy it into the FCB so that the file can be opened.

File Delete : CE2CH

Entry:

DE : FCB address

Return:

AF : Error code

This routine deleted the directory entry that belongs to the opened file.

Load Machine code program : CE30H

Entry:

DE : FCB address
in FCB : Filename

Return:

AF : Error code

This routine opens the indicated file and loads the machine code program into memory.

Execute Machine code program : CE33H

Entry:

DE : FCB address
in FCB : Filename

(does not return)

This routine opens the indicated file, loads the machine code program into memory and jumps to the entry address when no error occurred. If an error occurred, a DISK-Error message is given.

Enter EOF into Directory : CE51H

Entry:

DE : FCB address

Return:

AF : Error code

This routine enters an end of file into the directory of the floppy disk.

The routines mentioned before make full usage of files from machine code programs possible. There are however more routines available, that allow access to the floppy disk without using files. They should be used with care, since a wrong usage can destroy data on the floppy disk.

Drive Select : CE5BH

Entry:

A : Drive number

Return:

AF : Error code

This routine selects the drive for the all following read/write operations.

Drive Test : CE5EH

Entry:

A : Drive number

Return:

AF : Error code

This routine selects the drive A and tests, whether the drive is ready and if a floppy disk is inserted.

Read Sector : CF6FH

Entry:

DE : Sector number
HL : Buffer address

Return:

AF : Error code
in buffer: Data read

This routine reads one sector and stores the data in a buffer. The sector number is present in register pair DE; the buffer address in register pair HL.

Test Sector : CF73H

Entry:

DE : Sector number

Return:

AF : Error code

This routine test if a sector can be read.

Write Sector : CF7FH

Entry:

DE : Sector number
HL : Buffer address
in buffer: Data to be written

Return:

AF : Error code

This routine writes the data from a buffer to a floppy disk sector. The sector number is present in register pair DE; the buffer address in register pair HL.

Write Protected Sector : CF7BH

Entry:

DE : Sector number
HL : Buffer address
in buffer: Data to be written

Return:

AF : Error code

This routine writes the data from a buffer to a floppy disk sector. The sector number is present in register pair DE; the buffer address in register pair HL. The sector is written with a mark. This normally only happens with directory sectors. When such a sector is read, an error 6 occurs.

Re-select Drive : CE16H

Return:

AF : corrupted

This routine again selects the last selected drive.

Read Directory sector : D25FH

Entry:

A : Directory sector number

Return:

AF : Error code
HL : Buffer address (5900H)

This routine reads a directory sector (A contains the sector number) and stores it into the system buffer at 5900H.

Write Directory sector : D274H

Entry:

in system buffer : Data to be written

Return:

AF : Error code
HL : corrupted

This routine writes the directory sector last read back to the floppy disk.

One should take care using the write routines. A floppy disk with a damaged directory becomes unreadable in most cases.

DOS also contains additional routines, that are not related to the usage of floppy disks, but can also be useful.

Issue Error Message : CE90H

Entry:

A : Error code

This routine generates a DISK-Error message and returns to BASIC.

Multiply : CE76H

Entry:

HL : Multiplicand
A : Multiplier

Return:

AHL : Result

This routine performs a multiplication. The result has a length of 3 bytes maximum.

Division : CE79H

Entry:

HL : Dividend
A : Divisor

Return:

HL : Result
A : Fraction
BC : corrupted

Give Time : CE6DH

Entry:

HL : Buffer address

Return:

in buffer: The time as an 8 byte string
BC : corrupted
DE : corrupted
HL : corrupted

The time string uses the format hh:mm:ss.

Give Date : CE70H

Entry:

HL : Buffer address

Return:

in buffer: The date as an 8 byte string
BC : corrupted
DE : corrupted
HL : corrupted

The date stored in the memory locations 4044H - 4046H is put in the buffer using the format dd.mm.yy

Insert Interrupt routine : CE10H

Entry:

DE : Address of the routine to be inserted

Return:

DE : corrupted
HL : corrupted

This routine inserts a program present in memory into the interrupt chain. This call has the same effect as the CMD "Y address" statement in BASIC. The program must not already be a part of the interrupt chain! It may not corrupt any Z80 register. This DOS call also executes a EI instruction.

Remove Interrupt routine : CE13H

Entry:

DE : Address of the routine to be removed

Return:

BC : Corrupted
DE : Corrupted
HL : Corrupted

This routine removes a program from the interrupt chain. This call has the same effect as the CMD"V address" statement in BASIC. This DOS call also executes a EI instruction.

6. Interrupt routine format

On the Colour Genie, it is possible to insert routines in a so called interrupt chain. Every 25 milliseconds, the computer walks through this chain and looks if routines should be executed.
A routine in this chain must have the following header:

Address	Description
xxxx+0,xxxx+1	Vector to next interrupt routine (set by system)
xxxx+2	Initial timer value
xxxx+3	timer counter value
xxxx+4	Entry address of the interrupt routine.

Inserting the routine is done from BASIC by using the CMD"Y xxxx" statement or by executing DOS call CE10H with the address xxxx in register DE. The system then inserts it into the chain and stores the vector of the next interrupt routine in the header. The address xxxx+2 contains a initial timer value.
The timer counter value at xxxx+3 is decremented every timer tick of 25 ms. When it reaches zero, the interrupt routine is executed and the initial timer value on address xxxx+2 is copied to the timer counter value of xxxx+3.

The address xxxx+4 is the entry address of the interrupt routine, here is where the executable code starts. The routine must not corrupt any registers or the stack!

Note that the Z80 CPU must be in interrupt mode 1 (IM 1) to perform the executing of the interrupt chain.

7. Compatibility with TRS 80 / Video Genie Operating Systems

The most important routines in DOS are compatible with the corresponding routines of the most TRS80 / Video Genie Operating Systems. The following routines are present in all these operating systems:

Colour Genie	TRS 80 / Video Genie
0013H	0013H
001BH	001BH
CE20H	4420H
CE24H	4424H
CE36H	4436H
CE39H	4439H
CE3CH	443CH
CE42H	4442H
CE28H	4428H
CE2CH	442CH

Other routines are not present in all operating systems. A routine that starts at CEXXH on the Colour Genie, starts at 44XXH on the TRS 80 / Video Genie

The floppy disks are fully exchangeable. The formats for BASIC and machine code programs is identical.

8. Floppy disk formats

The Colour Genie DOS can handle the following floppy disks formats:

SS = Single Sided
DS = Double Sided

SD = Single Density
DD = Double Density

Sided	Density	Tracks	Sector/Track	Total Sectors	Capacity
SS	SD	40	10	400	102
SS	DD	40	18	720	184
SS	SD	80	10	800	204
SS	DD	80	18	1440	368
DS	SD	40	20	800	204
DS	DD	40	36	1440	368
DS	SD	80	20	1600	408
DS	DD	80	36	2880	736

In order to achieve the right format, the floppy disk drive has to be set to the proper drive type. This can be done with the CMD"< drive# = Type" statement in BASIC. The drive# indicates for which of he 4 drives (0..3) the new type applies. Default at start-up is drive type C.

Type	Floppy (Tracks)	Drive (Tracks)	Density	Sided
A	40	40	SD	SS
B	40	40	SD	DS
C	40	40	DD	SS
D	40	40	DD	DS
E	40	80	SD	SS
F	40	80	SD	DS
G	40	80	DD	SS
H	40	80	DD	DS
I	80	80	SD	SS
J	80	80	SD	DS
K	80	80	DD	SS
L	80	80	DD	DS

The types E, F, G and H are necessary for reading a 40 track floppy disk on a 80 track disk drive.

9. Reading alien floppy disks

It is possible for the Colour Genie to read floppy disks, that are made on the Genie I/II/III computers. Important are the Pdrive (see G-DOS manual) correspond to the settings of the Colour Disk BASIC, since no adaptation is possible from within the Colour Disk BASIC.

Following settings are required by the Colour Genie, so that alien floppy disks can be read:

Type	Type Interface	Type Drive	Units In Block	Number Dir. Units	Start Block Directory
A	A	A	2	2	20
B	A	C	4	4	20
C	CK	E	3	3	24
D	CK	G	6	6	24
E	AL	A	2	2	20
F	AL	C	4	4	20
G	CKL	E	3	3	24
H	CKL	G	6	6	24
I	A	A	2	2	40
J	A	C	4	4	40
K	CK	E	3	3	48
L	CHK	G	6	6	48

Floppy disks, made on the Colour Genie, can be read on the Genie I/II/III using the Pdrive settings above. The value for Start Block Directory however does not have to match.

Note that the Pdrive settings are not put on the floppy disks by the Colour Genie and a call of this table on a Genie I/II/III causes an error.

Appendix A: Disk BASIC Memory usage.

Disk BASIC needs parts of the BASIC system memory (4000H - 43FFH) and parts of the normal memory from 5800H to 5C9AH, followed by the buffers (0-9) needed by files that are used by BASIC programs. Then the memory for BASIC programs follows.

The following memory locations are important under DOS :

```
4040H              25 milliseconds counter for the time
4041H              Seconds counter
4042H              Minutes counter
4043H              Hours counter
4044H - 4046H      Date
4050H - 4056H      Interrupt routine for the time
4057H - 405DH      Interrupt routine for displaying the time
405EH              Number of last read directory sector (for write
                   routine at D274H)
4076H              Hash code of the last opened file
407CH - 407DH      Entry address for a program loaded with CMD"L"
408EH              Number of files in Disk BASIC
43CEH - 43E1H      Disk BASIC FCB addresses (2 byte pairs)
43E3H              Error code of last reported Disk Error
43ECH - 43FFH      USR functions entry addresses (2 bytes pairs)
5800H - 58FFH      Buffer for Close and Kill
5900H - 59FFH      System buffer for reading of directory
5A08H              Actual drive number
5A0AH - 5A13H      Actual drive data
5A71H - 5A98H      Data for drives 0 - 3
CEA0H - CF17H      Data for drive types A - L
```

Drive data are loaded from EPROM to 5A71H - 5A98H when a CMD"<" statement is executed. When a drive is selected, the drive data of the selected drive is copied to 5A0AH - 5A13H. This data block has to following contents :

```
5A0AH              Number of first lump belonging to directory
5A0BH              Number of lumps on floppy disk
5A0CH              Stepper motor speed: 7 for SD, 53H for DD
5A0DH              Number of tracks (40/80)
5A0EH              Number of sectors per track
5A0FH              Number of granules per lump
5A10H              Always 0
5A11H              Flags used for control of floppy disk controller
5A12H              Number of sectors per granule (always 5)
5A13H              Directory length in granules
```

Appendix B: Memory map

	Hex	Decimal	Peek/Poke
ROM Cartridge (1 KB)	FFFF	65535	-1
	FC00	64512	-1024
Keyboard Matrix (1 KB)	FBFF	64511	-1025
	F800	63488	-2048
User-definable Characters (1 KB)	F7FF	63487	-2049
	F400	62464	-3072
Character Colour Memory (1 KB)	F3FF	62463	-3073
	F000	61440	-4096
DOS Utilities (4 KB)	EFFF	61439	-4097
	E000	57344	-8192
DOS Routines (4¼ KB)	DFFF	57343	-8193
	CF00	52992	-12544
DOS Vectors (¼ KB)	CEFF	52991	-12545
	CE00	52736	-12800
BASIC Vectors (3½ KB)	CDFF	52735	-12801
	C000	49152	-16384
User RAM (EG 2011) (16 KB)	BFFF	49151	-16385
	8000	32768	-32768
User RAM (8 KB)	7FFF	32767	32767
	6000	24576	24576
Program Buffers 1 to 3 (1 at lowest address) (1½ KB)	5FFF	24575	24575
	5A00	23040	23040
I/O Buffer (DOS) (¼ KB)	59FF	23039	23039
	5900	22784	22784
Floppy Disk Buffer (¼ KB)	58FF	22783	22783
	5800	22528	22528
Graphic Screen Memory (4 KB)	57FF	22527	22527
	4800	18432	18432
Text Screen Memory (1 KB)	47FF	18431	18431
	4400	17408	17408
Vectors, Marks and System Variables (DOS and BASIC) (1 KB)	43FF	17407	17407
	4000	16384	16384
BASIC Interpreter (EPROM) (16 KB)	3FFF	16383	16383
	0000	0	0

Appendix C: Error codes

The following table contains the error codes and their meaning. This table is also valid for the G-DOS operating system (Genie I/II/III) and contains therefor some error codes that can not occur under the Colour Disk BASIC.

Code Decimal	Hex	Error description
0	00	No error
1	01	Bad file data
2	02	Read error: search error
3	03	Read error: data lost
4	04	Read error: checksum error
5	05	Read error: record not found
6	06	Read error: trying to read protected sectors
7	07	Read error: trying to read system sectors
8	08	Device unreachable
9	09	Undefined error code
10	0A	Write error: search error
11	0B	Write error: data lost
12	0C	Write error: checksum error
13	0D	Write error: record not found
14	0E	Write error on disk drive
15	0F	Floppy disk is write protected
16	10	Peripheral device unreachable
17	11	Directory read error
18	12	Directory write error
19	13	Illegal filename
20	14	Track number too high
21	15	Illegal DOS call function
22	16	Undefined error code
23	17	Undefined error code
24	18	File not in directory
25	19	Access denied by file
26	1A	Directory full
27	1B	Floppy disk full
28	1C	End of file reached
29	1D	Beyond end of file
30	1E	Directory full: cannot extend file
31	1F	File not found
32	20	Illegal or missing disk drive
33	21	No device reachable
34	22	Load error: bad format
35	23	Memory error
36	24	Try to load in ROM
37	25	Load error: access denied
38	26	File not open
39	27	Illegal initialisation data on floppy disk
40	28	Illegal track number
41	29	Illegal logical filenumber
42	2A	Illegal DOS function
43	2B	Illegal function under chaining
44	2C	Directory incorrect
45	2D	Bad FCB data
46	2E	System program not found
47	2F	Bad parameter

48	30	No filename
49	31	Bad floppy disk type
50	32	Read error BOOT
51	33	Fatal DOS error
52	34	Illegal abbreviation, separator or end marker
53	35	File already exists
54	36	Command too long
55	37	Access denied by floppy disk
56	38	Not a Mini-DOS function
57	39	Forced termination of function
58	3A	Difference at verify
59	3B	Insufficient memory
60	3C	Incompatible drive or floppy disk
61	3D	ADE=N attribute, cannot extend file
62	3E	Cannot extend file at read

Appendix D: System Entry Table

Address	Name	Description
CE00H	$DOS	Return to command mode
CE05H	$DOSCMD	Calls machine code program or DOS command
CE08H	$NERROR	RET Z, else
CE09H	$DERROR	Report error message
CE0DH	$DEBUG	Call system monitor
CE10H	$ENQUE	Insert interrupt routine
CE13H	$DEQUE	Remove interrupt routine
CE16H	$RESEL	Reselect current disk drive
CE19H	$DOSCAL	as $DOSCMD, but returns
CE1CH	$EXFIL	Examine and transfer filename into FCB
CE20H	$INIT	Initialises file and opens FCB
CE24H	$OPEN	Opens FCB for current file
CE28H	$CLOSE	Closes FCB
CE2CH	$KILL	Deletes file entry
CE30H	$LOAD	Loads machine code program
CE33H	$RUN	Runs machine code program
CE36H	$RDSEC	Reads sector
CE39H	$WRSEC	Writes sector
CE3CH	$WRSECV	Writes and verifies sector
CE3FH	$POS0	Set FCB sector number to zero
CE42H	$POSBC	Set FCB sector number to (BC)
CE45H	$POSDEC	Decrements FCB sector number
CE48H	$POSEOF	Sets FCB sector number at end of file
CE4BH	$ALLOC	Reserves unit on floppy disk for file
CE4EH	$POSBRA	Positions FCB at relative byte address
CE51H	$WREOF	Write end of file into directory
CE54H	$DELIM	Tests delimiters at parameters
CE5BH	$DRVSEL	Selects disk drive
CE5EH	$DSKMNT	Tests if floppy disk in drive
CE67H	$PRINT	Prints text on screen
CE6AH	$LPRINT	Prints text on printer
CE6DH	$CONTIM	Gives time as 8 bytes string
CE70H	$CONDAT	Gives date as 8 bytes string
CE76H	$MULT	Multiply A*HL=AHL
CE79H	$DIV	Divides HL div A=HL mod A
CE79H	$HEXDE	Gives hexadecimal representation of DE
CE80H	DOSFCB	System internal FCB of 32 byte length

Appendix E: Description of the Restart commands

The Restart commands (RST) are subroutines in Z80 machine language (like CALL routines). The advantage of RST commands is that they use little memory. A CALL command uses 3 bytes; a RST command uses only 1 byte. The following routines are used as subroutines in Colour Disk BASIC:

RST 00 — Jumps to memory location 0 and corresponds with a cold boot

RST 08 — Syntax check
The memory address that is addressed by HL is compared to the byte following the RST 08. If they are equal, a RST 10 is executed, else a syntax error is reported.

RST 10 — Loads the memory location addressed by HL+1 into the accumulator A.
Blanks and Linefeeds are skipped. Digits cause the setting of the Carry-flag, a ':' or a 00 cause the setting of the Zero-flag

RST 18 — Compares HL with DE register pair

HL > DE:	Zero-flag=0,	Carry-flag=0
HL = DE:	Zero-flag=1,	Carry-flag=0
HL < DE:	Zero-flag=0,	Carry-flag=1

RST 20 — Tests the type of the contents of the X-register stored at address 40AFH.

Integer:	Zero-flag=0,	Carry-flag=1,	Parity-Flag=1,	Sign-Flag=1
Single:	Zero-flag=0,	Carry-flag=1,	Parity-Flag=0,	Sign-Flag=0
Double:	Zero-flag=0,	Carry-flag=0,	Parity-Flag=1,	Sign-Flag=0
String:	Zero-flag=1,	Carry-flag=1,	Parity-Flag=1,	Sign-Flag=0

RST 28 — Used to load a system module. The Accumulator A contains the loader code. RST 28 is also called when pressing the BREAK key.

RST 30 — Generally used to call a DEBUG program. It returns after termination of the DEBUG program.

RST 38 — This routine is executed every 25 milliseconds if the Z80 CPU is set to interrupt mode 1 (IM 1)

Appendix F: Vector tables for DOS / Disk BASIC

The system variable space between 4000H and 43FFH contains the vectors for the Disk BASIC statements and the vectors into DOS. Every vectors requires 3 bytes (space necessary for a absolute jump instruction).

The first table occupies the area from 4152H to 41A5H. It contains the vectors for the Disk BASIC statements. Under Level II BASIC, these vectors all contain a jump to 013BH in the BASIC interpreter ROM area. Under Disk BASIC, the vectors contain a jump to the following addresses:

Address (Dec, Hex)		Jump to:	Statement
16722	4152H	C565H	CVI
16725	4155H	C3AEH	FN
16728	4158H	C562H	CVS
16731	415BH	C353H	DEF
16734	415EH	C55FH	CVD
16737	4161H	CA25H	EOF
16740	4164H	CA04H	LOC
16743	4167H	C9FFH	LOF
16746	416AH	C54EH	MKI$
16749	416DH	C54BH	MKS$
16752	4170H	C548H	MKD$
16755	4173H	C2FBH	CMD
16758	4176H	C24DH	TIME$
16761	4179H	CB5CH	OPEN
16764	417CH	C5D0H	FIELD
16767	417FH	CAA1H	GET
16770	4182H	CAA0H	PUT
16773	4185H	C9C4H	CLOSE
16776	4188H	C8D6H	LOAD
16779	418BH	C962H	MERGE
16782	418EH	CC0EH	NAME
16785	4191H	CBEBH	KILL
16788	4194H	1E4AH	none (Previous &)
16791	4197H	C57EH	LSET
16794	417AH	C57FH	RSET
16797	419DH	C456H	INSTR
16800	41A0H	C99CH	SAVE
16803	41A3H	C65BH	LINE

The second table contains the BASIC vectors into DOS. Under Level II BASIC, these vectors contain a return instruction followed by 2 nops to fill the remaining 2 bytes memory space reserved for each vector. Under Disk BASIC, the vectors contain a jump to the following addresses:

Address (Dec, Hex)		Jump to:	Description
16806	41A6H	C1D3H	Error routine
16809	41A9H	C390H	USR
16812	41ACH	C959H	Return to active command mode
16815	41AFH	C6BBH	Input line
16818	41B2H	C98BH	After tokenizing
16821	41B5H	C75FH	After accepting a new program line
16824	41B8H	C76EH	After accepting a new program line
16827	41BBH	C9F5H	Clear after cleaning all variables
16830	41BEH	C685H	After ending printer output
16833	41C1H	C6F2H	Character output
16836	41C4H	C701H	Keyboard read at program execution
16839	41C7H	C8D3H	RUN
16842	41CAH	C62AH	PRINT
16845	41CDH	C77CH	PRINT (numeric value)
16848	41D0H	C77BH	Start of a new program line
16851	41D3H	C747H	PRINT or PRINTTAB
16854	41D6H	C68DH	INPUT
16857	41D9H	C4DAH	MID$ left of = sign
16860	41DCH	C7C2H	Data processing after READ / INPUT
16863	41DFH	C6A5H	Termination of INPUT
16866	41E2H	CC16H	SYSTEM

Appendix G: Program examples

Error report with return to program:

```
ERROR       PUSH  HL
            PUSH  AF
            CALL  2169H       ;Output to screen
            CALL  20F9H       ;Cursor to start of line
            LD    HL,TEXT1
            CALL  2B75H       ;Output text
            POP   AF
            LD    L,A
            LD    H,0         ;Error code to HL
            CALL  0FAFH       ;Output HL
            CALL  20FEH       ;Output CR
            POP   HL
            OR    FFH         ;Clear Z flag
            RET
TEXT1       DEFM  'Disk-Error '
            DEFB  0
```

Open file with request for filename (ROPEN for read, WOPEN for write):

```
TEXT2       DEFM  'Filename: '
            DEFB  0
OPEN        LD    HL,TEXT2
            CALL  2B75H       ;Output text
            PUSH  BC          ;Save record length
            LD    HL,5B08H    ;Buffer address
            LD    B,18H       ;Maximum length of input
            CALL  5D9H        ;Input filename
            LD    C,B
            LD    B,0
            PUSH  HL
            ADD   HL,BC
            LD    (HL),0      ;End marker
            POP   HL
            LD    DE,FCB      ;FCB address
            CALL  0CE1CH      ;Copy filename
            CALL  NZ,ERROR    ;Call if error
            POP   BC          ;Restore record length
            JR    NZ,OPEN     ;Input new filename
            LD    HL,BUFFER   ;256 bytes buffer
            CALL  0CE24H      ;Open file
            RET   Z           ;If no error return
            CP    18H         ;Test error code
            CALL  NZ,ERROR    ;Report error
            JR    NZ,OPEN     ;Enter new filename
            OR    A           ;File not found
            RET
ROPEN       CALL  OPEN        ;Open file
            RET   Z           ;If no error return
            CALL  ERROR       ;Report error
            JR    ROPEN       ;Try again
WOPEN       CALL  OPEN        ;Open file
            JR    Z,EXI       ;If file already exists jump
            CALL  0D694H      ;Create file
            RET   Z           ;If no error return
```

```
                CALL  ERROR       ;Report error
                JR    WOPEN       ;Try again
EXI             LD    HL,TEXT3
                CALL  2B75H       ;Ouput text
                CALL  384H        ;Wait for key pressed
                CP    'Y'
                RET   Z           ;If file may be used return
                JR    WOPEN       ;Try again
TEXT3           DEFM  'File already exists.'
                DEFB  0DH
                DEFM  'Use anyway?'
                DEFW  0DH
```

Close file:

```
CLOSE           LD    DE,FCB      ;FCB address
                CALL  0CE28H      ;Close file
                CALL  NZ,ERROR    ;If failure report error
                JP    NZ,PROG     ;And jump to start of program
                RET
```

Write byte:

```
WBYTE           PUSH  DE          ;Save registers
                PUSH  AF
                LD    DE,FCB      ;FCB address
                CALL  1BH         ;Write byte
                CALL  NZ,ERROR
                JR    NZ,PROG
                POP   AF
                POP   DE
                RET
```

Read byte (Carry flag is set at end of file):

```
RBYTE           PUSH  DE          ;Save register
                LD    DE,FCB
                CALL  13H         ;Read byte
                JR    Z,NOERR     ;If no error jump
                CP    1CH         ;EOF code
                CALL  NZ,ERROR    ;If not EOF report error
                JP    NZ,PROG
                SCF
                JR    EOF
NOERR           OR    A           ;Clear carry flag
EOF             POP   DE
                RET
```

Write a memory block from BSTART to BEND:

```
SAVE            LD    B,0         ;Sector operation
                CALL  WOPEN       ;Open file
                LD    HL,BSTART-1
                LD    DE,BEND
LOOP            INC   HL
                LD    A,(HL)      ;Byte into A
                CALL  WBYTE       ;Write byte
                RST   18H         ;Compare HL and DE
```

```
                JR    NZ,LOOP     ;Loop if not equal
                JP    CLOSE       ;Close file
```

Load a memory block:

```
LOAD            LD    B,0         ;Sector operation
                CALL  ROPEN       ;Open file
                LD    HL,BSTART
                LD    DE,BEND
LOOP1           CALL  RBYTE       ;Read byte
                RET   C           ;If EOF then return
                LD    (HL),A      ;Store byte
                INC   HL
                JR    LOOP1       ;Next byte
```

The next two routines make it possible to use an opened file as a memory of 64 KByte at maximum. READ corresponds with LD A,(HL) and WRITE corresponds with LD (HL),A .

```
SETPOS          PUSH  BC
                PUSH  HL
                LD    C,L         ;Address to HLC
                LD    L,H
                LD    H,0
                CALL  0CE4EH      ;Set position
                POP   HL
                POP   BC
                RET
READ            CALL  SETPOS      ;Set position
                JR    RBYTE       ;Read byte to A
WRITE           CALL  SETPOS      ;Set position
                JR    WBYTE       ;Write byte
```

Write records that contain a maximum of 20 characters, read from the keyboard:

```
WRITE1          LD    B,20        ;Record length
                CALL  WOPEN       ;Open file
LOOP2           LD    HL,BUFFE1   ;20 byte buffer
                LD    B,20        ;Maximum length
                PUSH  HL
                CALL  5D9H        ;Input from keyboard into buffer
                POP   HL
                JP    C,CLOSE     ;If BREAK key pressed jump
                LD    DE,FCB      ;FCB address
                CALL  0CE39H      ;Write record
                CALL  NZ,ERROR
                JP    NZ,PROG
                JR    LOOP2       ;Next record
```

Read the fifth record of the file created above:

```
READ1           LD    B,20        ;Record length
                CALL  ROPEN       ;Open file
                LD    DE,FCB      ;FCB address
                LD    BC,4        ;Number of fifth record
                CALL  0CE42H      ;Set position
                CALL  NZ,ERROR
                JP    NZ,PROG
                LD    HL,BUFFE1   ;20 byte buffer
```

```
                CALL  0CE36H      ;Read record
                CALL  NZ,ERROR
                JP    NZ,PROG
                RET
```

Calculate the sector number of the first directory sector of the floppy disk in drive 0:

```
CALC            LD    A,0         ;Drive number
                CALL  0CE5EH      ;Test drive
                CALL  NZ,ERROR
                JP    NZ,PROG
                LD    HL,0        ;Sector number
                LD    DE;BUFFE2   ;256 byte buffer
                CALL  0CF6FH      ;Read sector 0
                CALL  NZ,ERROR
                JP    NZ,PROG
                LD    A,(BUFFE2+2);Third byte into A
                LD    L,A
                LD    H,0
                LD    A,(5A0FH)   ;Granules per lump
                CALL  0CE76H      ;Multiplication
                LD    A,(5A12H)   ;Sectors per granule
                CALL  0CE76H      ;Multiplication
                RET               ;Result in HL
```

Calculate the number of free granules of the floppy disk in drive 0:

```
CALC1           LD    A,0         ;Drive number
                CALL  0CE5EH      ;Test drive
                CALL  NZ,ERROR
                JP    NZ,PROG
                XOR   A           ;A=0
                CALL  0D25FH      ;Read directory sector 0
                CALL  NZ,ERROR
                JP    NZ,PROG
                LD    IX,0        ;Set counter on 0
                LD    A,(5A0FH)   ;Granules per lump
                LD    D,A
                LD    A,(5A0BH)   ;Lumps on the floppy disk
                LD    E,A
                LD    HL,5900H    ;Address of system buffer
LOOP3           LD    A,(HL)      ;Read byte into A
                LD    B,D
LOOP4           RRA               ;Bit 0 into carry
                JR    C,NTFREE    ;If granule not free jump
                INC   IX          ;Increment counter
NTFREE          DJNZ  LOOP4
                INC   HL          ;Next byte
                DEC   E
                JR    NZ,LOOP3    ;Loop for every lump
                RET               ;Result in IX
```

Change the name of the floppy disk:

```
NEWNAM          LD    A,0         ;Drive number
                CALL  0CE5EH      ;Test drive
                CALL  NZ,ERROR
                JP    NZ,PROG
```

```
                XOR   A
                CALL  0D25FH      ;Read directory sector 0
                CALL  NZ,ERROR
                JP    NZ,PROG
                LD    DE,59D0H    ;Address of floppy disk name in
                                  ;buffer
                LD    HL,NAME     ;Address of the new name
                LD    BC,8        ;Length of the name
                LDIR              ;Replace old with new name
                CALL  0D274H      ;Write directory sector 0 back to
                                  ;the floppy disk
                CALL  NZ,ERROR
                JP    NZ,PROG
                RET
```

Insertion of an interrupt routine (the routine beeps every 2 seconds):

```
       PING     LD    DE,INT      ;Address of interrupt routine
                PUSH  DE
                CALL  0CE13H      ;Just to be safe the routine is
                                  ;first removed from
                                  ;the interrupt chain
                POP   DE
                CALL  0CE10H      ;Insert interrupt routine
                RET
       INT      DEFW  0           ;Storage space for interrupt
                                  ;vector
                DEFB  80          ;Starting value for counter
                                  ;80*25ms
                DEFB  80          ;Counter
                PUSH  AF          ;Save registers
                PUSH  BC
                PUSH  DE
                PUSH  HL
                CALL  357CH       ;Beep
                POP   HL          ;Restore registers
                POP   DE
                POP   BC
                POP   AF
                RET
```